MACHINE LEARNING ALGORITHM OF DETECTION OF DOS ATTACKS ON AN AUTOMOTIVE TELEMATIC UNIT

Eric Perraud

Renault Software Labs, Toulouse, France

ABSTRACT

Today vehicles are connected to private networks which are owned by the car manufacturer. But in coming years, vehicles become more and more connected to the public Internet for infotainment applications but also to safety applications. Like any Internet terminal, some hackers can attack the wireless connectivity unit of the vehicle with Distribution Denial of Services (DDOS) attacks, so that the wireless connectivity unit of the vehicle is not available and the service is lost. Therefore, it is critical to developing a mechanism to detect such an attack and eliminate it, to maintain the availability of the wireless connectivity unit. This paper proposes an algorithm which proceeds in 2 steps: it uses an unsupervised machine learning algorithm to detect DDOS attacks in the incoming Internet data. When it detects an attack, it uses the results of the machine learning algorithm to split the legitimate flow and the rogue flows. The rogue flow is filtered so that the availability of the wireless connectivity unit of the vehicle is restored. This proposed algorithm needs very few CPU computing power and is compatible with low-cost CPUs which are used in an automotive wireless connectivity unit.

KEYWORDS

Clustering algorithm, vehicle, DDOS, unsupervised learning

1. INTRODUCTION

Vehicles become more and more connected, with wireless radio technologies (4G, Wifi…). As of today, the vehicles are connected to the car manufacturer network and more recently to road operator networks (thanks to the V2X radio connectivity). In the mid-term, the vehicles are connected to the public Internet for infotainment applications or to service providers which offer services for driving assistance (like real-time HD-maps, road safety applications …). Then the vehicles become Internet terminals and therefore they are subject to cyber-attacks, particularly DDOS attacks. For the latter applications which are safety-related, it is critical to keep the wireless connectivity available and protect it against DDOS attacks. At least, it is key to detect when the in-vehicle connectivity unit (IVC-U) is not available because of a DDOS attack so that the onboard software can take appropriate back-up actions. Ideally, it is even better if the DDOS attacks can be filtered out and the IVC-U availability is restored so that the vehicle can access desired off-board services.

Skilled network administrators protect their networks and Internet devices against DDOS attacks by monitoring the incoming data traffic and comparing some traffic patterns with prior known attack signatures ([9]) or simulated attack signatures (as explained in [10]). But the attacks evolve at every day (zero-day attack) and attack signature databases should be continuously updated. It is not an acceptable solution for an IVC-U. The only viable solution is an automated mechanism with unsupervised learning.

Some detectors of suspect incoming traffic have recently emerged using unsupervised learning ([3], [4], [6], [8], [9]). They use machine learning techniques. They rely on the below observations:

- Most of the incoming Internet traffic is the legitimate traffic whereas the rogue traffic is an infrequent IP flow and has a low occurrence rate [17] [4] [6],
- When using pertinent features, the legitimate traffic appears as data points which converge into clusters whereas the data traffic anomalies appear as outlier points [12] [13] [4] [6] [17].
-

However, such machine learning algorithms are typically very computing-intensive and require a lot of computing power. The algorithms which are disclosed in the literature are not compatible with low-cost MCUs of IVC-U. Therefore, any algorithm of detecting and filtering DDOS attacks in an IVC-U shall fulfill the following requirements:

- Monitoring the incoming Internet traffic in the IVC-U when there is no DDOS attack, shall consume very few CPU resources and shall not degrade the quality of service of the wireless connectivity (particularly of the safety-critical services),
- The detection time shall be quite fast, excluding any algorithm whose convergence time is very long (in the range of seconds),
- The needed peak computing power to detect a DDOS attack and filter out the DDOS flows shall be reasonable and compatible with the computing power of an IVC-U.

The algorithm which is proposed in this paper fulfills these requirements while providing a very good false alarm rate (none has been detected). The DDOS detection success is also pretty good: all the TCP SYN attacks have been detected.

2. Prior Art

In the literature, we can find multiple unsupervised algorithms of detection of anomaly or attacks in the incoming Internet data. They all try to determine clusters of data points which characterize the legitimate traffic. K-means [1] is a well-known clustering algorithm. But it suffers from prohibitive drawbacks for an IVC-U: it is long to converge because it is an iterative process, the number of clusters shall be known and the clusters depend on the initialization of the cluster center points [1] [16]. Other clustering techniques have been proposed. Ippoliti & all [2] proposed to use an online One-Class Support Vector Machines (SVM) algorithm: it computes a hyperplane which identifies a small region in the n-feature space where the feature data of legitimate traffic are located. This approach suffers from a major drawback for an automotive implementation: computing a hyperplane is quite expensive from a CPU-power perspective (this dimensionality issue is illustrated in [11]). Similarly, Vipin Das & all [15] have proposed to use SVM technique (combined with Rough Set theory to reduce the number of the feature) to detect malicious intrusion pattern: but it needs to be trained and again computing a hyperplane with many features is prohibitive for an IVC-U. Another family of unsupervised attack detection algorithms uses Principal Component Analysis (PCA) [1] [3], considering the attack as an anomaly in the received data stream. But again, it consumes quite a lot of CPU resources since it has to find the eigenvalues and the eigenvectors of the covariance matrix of the observations vectors of the incoming Internet traffic. Authors of [4], [5], [6] [14] and [18] have focused on developing algorithms which minimize the needs for computing power. It is called Orunada (Online and Real-time Unsupervised Network Anomaly Detection and Analysis). It resolves key blocking points of previous algorithms for an automotive implementation. For n features which characterize the incoming IP traffic into the IVC-U, the algorithm projects the input vector on $n * (n-1)/2$ 2D-subspaces and searches data clusters in these 2D-subspaces. By this means, it resolves the dimensionality issue [7] and the search for data cluster per subspace converges. Indeed, each subspace is a digital 2D-grid. A coarse resolution is good enough for our purposes because it is assumed that the incoming Internet data with DDOS attacks is projected on a cell of the grid which is far away from the clusters of legitimate traffic. It might be not true for all the subspaces (depending on the type of DDOS attacks and the considered features). But it should be true for some subspaces so that the DDOS attack can be detected. Another nice advantage of this algorithm is that it uses a sliding time as an observation period and is incremental: in most of the time (when there is no DDOS attack), the new vector of features of incoming IP data is very close to the old vector which leaves the observation window, so that the algorithm does not have to compute again the clusters of the 2D-subspaces (or in very few subspaces). However, the Orunada algorithm is designed for Internet network devices and not for an automotive wireless terminal. As demonstrated in [4], the processing time is about 50% of the CPU bandwidth, with 15 features to characterize the IP traffic, an acquisition time slot of 0.5s and with an Intel core i5 @2GHz. The CPU of an IVC-U is much less powerful than an Intel core i5 and the IVC-U can't assign so many CPU resources for monitoring DDOS attacks. That is the reason why the algorithm can't be used as is for an IVC-U. In addition, the internet traffic in network equipment has statistical properties and data patterns which are very different than those of an IVC-U. The number of open IVC-U connections at a given time is small but can change quite significantly in a short period of time whereas the relative variability of the number of connections in network equipment does not change so fast. As well, the incoming data traffic in an IVC-U can be very bursty with periods of no data. A consequence of that is the feature normalization: we can't know à-priori the min-max value of a feature because its range can change quite significantly between 2 acquisitions, depending on the nature of open connections. Let us consider the case where the vehicle uploads car data for maintenance purposes and then it starts to download big files of data: the incoming data flow is firstly composed of small TCP acknowledgement packets (about 60 bytes typically) and then the incoming data flow is composed of a mix of small TCP acknowledgement packets and large TCP data packets (a few Kbytes). If the average packet size was normalized with the max value of the observation window, it would be truncated during the transient period and would not be useful. As well, the Orunada algorithm considers one feature vector per IP flow. But for an IVC-U, there would typically be a few vectors per time slot because there are very few connections. When a DDOS happens, there would be N_{ddos} vectors of DDOS data where N_{ddos} is the number of DDOS spoofed addresses (usually very high) so that the DDOS feature vectors would cluster in a highly dense area and could be interpreted as legitimate traffic. For all these reasons, the Orunada algorithm shall be reworked to be used in an IVC-U.

3. Description Of The Proposed Algorithm

Figure 1 gives the context of the problem which is addressed in this paper. While the vehicle is connected to Internet services thanks to the IVC-U, some hackers spoof legitimate IP addresses and attempt to exhaust the IVC-U resources by DDOS attacks so that the IVC-U is not available for any additional service which the vehicle would like to get.

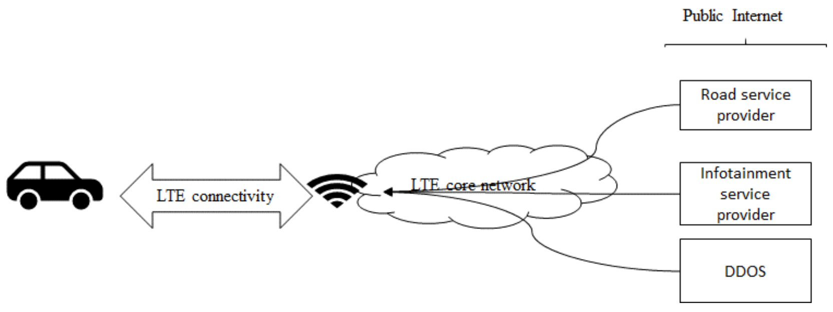

Figure 1. The vehicle is connected to Internet services and is also submitted to DDOS attacks

3.1 Input Vector and observation period

The algorithm collects the whole Internet incoming data on the IVC-U to build a single vector of IP features. The acquisition time slot dt (also called micro-slot in this paper) is at least 100ms and long enough to grab at least 100 Internet incoming packets. It does not distinguish the incoming flow per connections unlike [4]. The input vector is composed of 5 features:

$$X_n = (x_1^n, x_2^n, x_3^n, x_4^n, x_5^n) \tag{E1}$$

With:

x_1^n being the percentage of received TCP SYN packets over the dt time slot,
x_2^n being the percentage of received and transmitted ICMP packets over the dt time slot,
x_3^n being the average size of the received packets over the dt time slot,
x_4^n being the number of the source IP addresses the dt time slot,
x_5^n being the number of IP subnets (or IP@/24) over the dt time slot.

We do not consider the received bitrate or packet rate as some papers suggest [4, 5, 6] because the IVC-U is a wireless terminal. The received bitrate may vary in a very large scale because of radio link quality or because the cellular cell is heavily loaded. That is the reason why it has not been considered as a good feature to track some anomalies in the incoming flow. Y_n is the normalized vector of X_n:

$$Y_n = (y_1^n, y_2^n, y_3^n, y_4^n, y_5^n) \tag{E2}$$

x_i^n, i = 3…n, has a scalability problem: as a matter of fact, the observation window can't be used to predict the range of these features of the new input vector (like in [4]), particularly when the IVC-U opens or closes a connection. It is not the case for network equipment where the high number of open connections allows to accurately predicting the range of each feature thanks to the recent history. Therefore, we use a hardcoded upper bound for these 3 features to normalize them.

$y_1^n = x_1^n$,
$y_2^n = x_2^n$,
$y_3^n = \min(1, x_3^n / maxPktSize)$, with maxPktSize set at 1500 bytes assuming IPV4 traffic,
$y_4^n = \min(1, x_4^n / maxNbConnections)$, with maxNbConnections set at 10,
$y_5^n = \min(1, x_5^n / maxNbConnections)$.
Indeed, the weight of each feature for detecting a DDOS attack is done when assessing the abnormality score of the new input vector, as explained in the next paragraph.

An observation period is a sliding window which consists of the N latest time slots of legitimate data traffic, each time slot being characterized by E2.

$$A' = (Y_{n-N}, \dots . Y_{n-1}) \tag{E3}$$

A' is a matrix which characterizes the legitimate incoming data traffic in the past N micro-slots (' is the transpose matrix operator).

If the algorithm detects that X_n suffers from DDOS attacks, it does not insert this new vector in the sliding observation window. The matrix A' is not updated while the DDOS attackers are not filtered out.

3.2 Clustering method

As demonstrated in [4], [6], [9] or [19], it is difficult to determine data clusters in high-dimension feature space: clusters might be sunk in noise or the convergence of clustering algorithms is a crucial issue. Authors have suggested to breakdown the problem in simpler problems: instead of searching data clusters in the high-dimension space of features, it is simpler to consider 2D-subspaces, to project the data points on all the 2D-subspaces and search clusters in these 2D-subspaces. As well, authors of [4], [5] and [6] have shown that, when selecting adequate features, the number of clusters per subspace is very small while the Internet data traffic is composed of legitimate data traffic. They have proposed to describe a 2D-subspace as a digital grid and to describe a cluster as set of cells of this digital grid. In our implementation, the grid of each subspace is a 10x10 grid. The grid resolution is a trade-off between 2 conflicting objectives:

- If the grid resolution is too coarse, some outliers (which would be due to data traffic anomalies) could vanish in cells of legitimate data traffic.
- If the grid resolution is too high, the density threshold (as specified later in this document) is more complex to set. As well, the data points of legitimate traffic may be scattered in too many cells. This might result in frequent clustering steps and thus it might impact the CPU load.

As a rule of thumb, we set the rule that the data points of the legitimate incoming data traffic on an IVC-U are scattered in less than 10 cells per subspace. We found that a 10x10 grid fulfills this requirement for typical IVC-U incoming data traffic.

Each cell k of the lth subspace has a density function $d_l(k)$, which is the number of vectors of the observation window, whose projection in the lth subspace is in the cell k, with $l = 1 \ldots N_{subspaces} = n*(n-1)/2$ (= 10 in our implementation). For the subspace l composed with the features (i, j), $\overrightarrow{Y_l^n}$ is the projected feature vector Y_n, at time slot n, in the lth subspace:

$$\overrightarrow{Y_l^n} = \left(x\_p_l^n = 10 * y_i^n , y\_p_l^n = 10 * y_j^n \right) \quad \text{(E4)}$$

The projection itself is a very simple operation. It consists of determining the cell index of $\overrightarrow{Y_l^n}$ in the subspace l:

$$Cell_l^n = (x\_p_l^n + 10 * y\_p_l^n) \quad \text{(E5)}$$

When the vector Y_n is not considered as abnormal, the sliding observation window is updated: the recent Y_n vector is appended in the A matrix and the oldest vector (Y_{n-N}) leaves the A matrix. Therefore d_l $(Cell_l^{n-N})$ is decremented and d_l $(Cell_l^n)$ is incremented, with $l = 1\ldots N_{subspaces}$. In most of the time, the cell density remains unchanged for most of the subspaces as shown in the experimental chapter. Let us consider S_l as the set of the subspaces where the new vector is not projected in the same cell as the oldest vector of the observation window.

$$S_l = \{ l = 1 \ldots N_{subspaces} | Cell_l^n \neq Cell_l^{n-N} \} \quad \text{(E6)}$$

A cell k is dense if d (k) is higher than a density threshold called Th_{dens}. In our implementation, the density threshold is set at 5. Like the grid resolution, the density threshold is a trade-off between 2 conflicting objectives:

- If the threshold is too high, many cells with data points of legitimate traffic are not dense. As a consequence, the range of the abnormality score between a data point of legitimate traffic and a data point of corrupted traffic is reduced.
- If the threshold is too low, most of the cells with data points are dense and a new data point of corrupted data traffic might be projected in a dense cell for some sub-spaces. It decreases the probability of detecting a DDOS attack.
-

As a rule of thumb, we set the rule that 90% the data points of the legitimate incoming data traffic on an IVC-U are projected in dense cells. We found that a density threshold set at 5% fulfills this requirement: if the observation window size is 200 micro-slots, the density threshold is then set at 10 data points. This threshold is correlated with the grid resolution.

For the subspaces $l \in S_l$, the list of dense cells is updated. Let us consider U_l the set of subspaces where the list of dense cells has changed after being updated:

$$U_l = \{ l \in S_l \, | list\ of\ dense\ cells\ at\ t = n\ is\ different\ of\ list\ of\ dense\ cells\ at\ t = n-1\} \quad (E7)$$

A cluster is a set of dense cells which have a neighbor cell in this set of cells. So, the clustering is redone only for subspaces whose index belongs to U_l. The clustering phase is the most computation intensive. But in most of the time (when there is no DDOS attack on the IVC-U), U_l is empty or it has very subspaces to be re-clustered, as it will be shown with experimental results of paragraph 5. That is the reason why this algorithm does not need very powerful processors and is compatible with the IVC-U computing power. It is also important to note that except for the feature normalization step, all the software processing does not need any floating-point computation and consists of handling software pointers and array indexes.

3.3 Abnormality score

The key idea is that the legitimate traffic in the observation window is characterized by cluster patterns in the $N_{subspaces}$ 2D-subspaces and that the incoming traffic is compared to the characterization of the legitimate traffic in the recent observation window. The new vector is inserted in the observation window if it is not suspect. For this purpose, the algorithm computes an abnormality score for every new input vector which determines how much the new input vector is different from the observation window. If the score is below a given threshold $Th_{detection}$, it is inserted in the A matrix of the observation window and the partition of clusters is recomputed as explained in the 3.2 paragraph, else it is considered as a suspect vector and counter-measures must filter out the DDOS attack. $Th_{detection}$ has been set after multiple experiments, as explained in section 5. The abnormality score is a global score over all the 2D-subspaces so that it can detect different types of DDOS attacks.

$$AbnormalScore(Y_n) = \sum_{l=1}^{N_{subspace}} Score\left(\overrightarrow{Y_l^n}\right) \quad (E8)$$

$Score\left(\overrightarrow{Y_l^n}\right)$ is the abnormality score of the vector Y_n when it is projected on the l^{th} subspace and shown in E9.

$$Score\left(\overrightarrow{Y_l^n}\right) = \frac{EuclidianDistance\left(\overrightarrow{Y_n^l}, ClosestCluster\#l\right)}{AvgAnomalyLevel(l)} \tag{E9}$$

If the projection of the input vector on the l^th subspace is inside a cluster, its score is null. If it is not, it depends on the distance of the vector to the closest cluster of the l^th subspace. The more far away the vector is, the more abnormal it is. It is scaled by the average abnormality level of the l^th subspace, which is computed according to E10.

$$AvgAnomalyLevel(l) = \sum_{j \in NonClusterCellList(subspace\#l)} EuclidianDistance(cell\#j, ClosestCluster\#l) * d_l(j) \tag{E10}$$

The scaling is needed for two reasons. It allows to fairly balance the scores per subspace when assessing the global abnormality score and to compensate the fact that the features are not adequately scaled when the vector Y_n is built. It also better measures the abnormality level in a given subspace: if all the data traffic during the observation window is inside clusters for this subspace, and if the new projected vector is out the clusters for this subspace, the input vector is very abnormal in this subspace (even if the distance to the closest cluster is not so large). In the other way, if many points of the observation window are out of clusters for this subspace, and if the new projected vector is out the clusters for this subspace, the input vector is not so abnormal.

When the input vector is inserted in the observation window, the average anomaly level has to be recomputed for the subspaces where the new vector and the old vector do not project in the same cell.

When a new connection is open or when an existing connection is closed, the statistics of the features, x_i^n,i = 3…n may drift significantly, resulting in a peak increase of the abnormality score. It is key that the abnormality score of the 1st input vector following this IVC-U transient behavior is far less than the abnormality score of a DDOS attack to avoid a false positive. The settings of the algorithm shall be set accordingly as shown in the next paragraph. When the input vector corresponding to a change of connection status is considered as safe and inserted in the matrix A of the observation window, it may cause a drift of the clusters or a size change of the clusters. But the abnormality score of the next input vectors quickly drops to a low level, as shown in the paragraph of experimental results.

3.4 Identification and filtering of DDOS flows

When the new input vector is detected as suspect, the algorithm starts a new step. First, it determines the most discriminant subspace: it is the subspace with the highest score in the global abnormality score. Then in a 2nd step, the input vector $\overrightarrow{Y_l^n}$ (l being the subspace where the DDOS attack has the highest abnormality score) is split in 2D-vectors $\overrightarrow{Z_j^n}$, each corresponding to an IP source address, with j = 1 …$Nb_{IpSrcAddr}$, $Nb_{IpSrcAdd}$ being the number of IP source addresses in the time slot dt = n. The $\overrightarrow{Z_j^n}$ vectors are the components of each incoming IP flow to $\overrightarrow{Y_l^n}$. These new vectors are projected on the most discriminant subspace. The algorithm computes a score for every vector $\overrightarrow{Z_j^n}$: if it is higher than a predefined threshold Th_{filter}, the IP address is considered as a rogue address and tagged as a DDOS flow. To filter out this flow, the algorithm discards this flow by writing it in the blacklist of the IP tables. Th_{filter} has been set by simulations and confirmed by experiments.

3.5 Flow chart

The overall algorithm is summarized by the flow chart of figure 2. For illustrative purposes, it is shown where the above equations are computed.

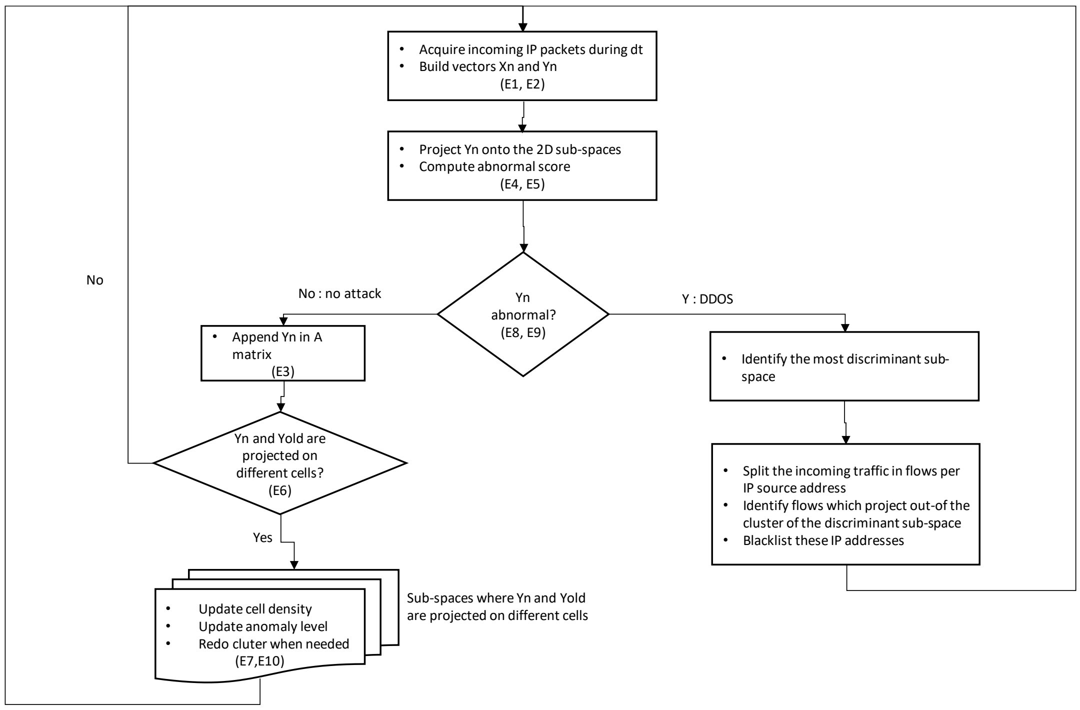
• Acquire incoming IP packets during dt
• Build vectors Xn and Yn
(E1, E2)
• Project Yn onto the 2D sub-spaces
• Compute abnormal score
(E4, E5)
Yn abnormal? (E8, E9)
No : no attack
Y : DDOS
No
• Append Yn in A matrix (E3)
Yn and Yold are projected on different cells? (E6)
Yes
• Update cell density
• Update anomaly level
• Redo cluter when needed
(E7,E10)
Sub-spaces where Yn and Yold are projected on different cells
• Identify the most discriminant sub-space
• Split the incoming traffic in flows per IP source address
• Identify flows which project out-of the cluster of the discriminant sub-space
• Blacklist these IP addresses

Figure 2. Flow chart

4. Simulation Results

4.1 Setup and Used data set

A MATLAB model is developed to validate the proposed algorithm. The data set consists of a Mawilab network trace as proposed in [4], captured on the 30th January 2015. We searched an IP destination which suffers from SYN flood attacks. The trace is shrunk to the traffic towards this IP address. It is our initial data set. It is then split into 2 data sets: a 1st data set without any SYN flood attack and a 2nd data set with SYN flood attack.

4.2 Simulation results

Figure 3 shows what the subspaces look like when there is no DDOS attack. Each subspace is a 10x10 grid. The indexes on the x and y axis show the centers of the cells. The color code is the following:

- Yellow indicates a dense cell,
- Blue indicates a non-dense cell with at least one projected vector: the blue color becomes darker with a higher number of data points,
- Very dark blue indicates a cell without any projected data point.

Note that the simulations are done with x_5^n = Time To Live (TTL) instead of number of the IP subnets.

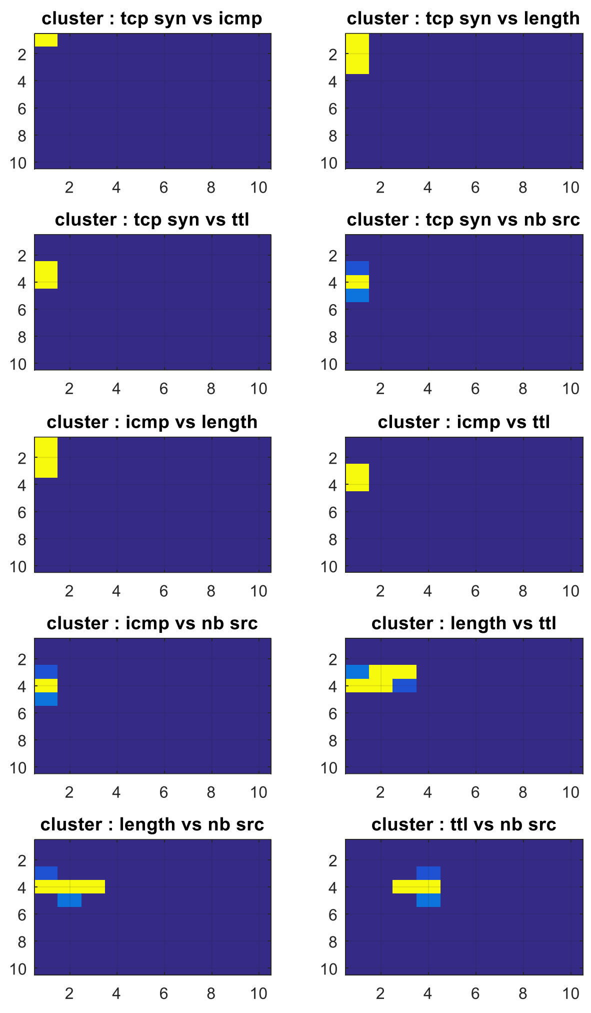

Figure3. Partitioning of 10 subspaces when there is no DDOS in the incoming data traffic

These simulations results are based on a real data set and validate the assumption that the legitimate incoming traffic converges towards dense clusters. The simulations also show that clusters of a few subspaces (the subspaces with the y_3^n or y_4^n features) breathe: a few cells alternate between dense and non-dense status so that the shape of the cluster looks breathing.

Figure 4 shows the subspaces when an input vector with a DDOS attack occurs. The color code is:

- Brown indicates the cell where the new input vector with DDOS attack is projected,
- Yellow indicates a dense cell,
- Blue indicates a non-dense cell with at least one projected vector,
- Very dark blue indicates a cell without any projected data point.

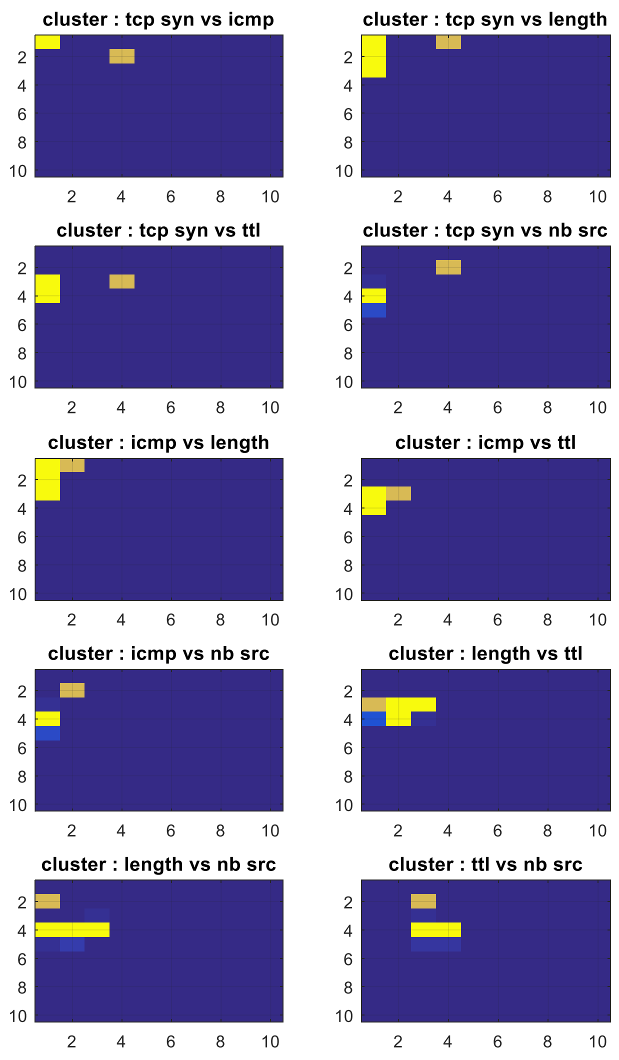

Figure 4. Projection of the incoming traffic (brown cell) when there is a DDOS attack

Figure 4 shows that in this example, the presence of the DDOS attack results in a pattern which is highly different from the pattern of the observation window, in 4 subspaces. It is then possible to detect it.

Figure 5 shows the abnormality score which is simulated with both data sets. The DDOS attack occurs at the 20th micro-slot.

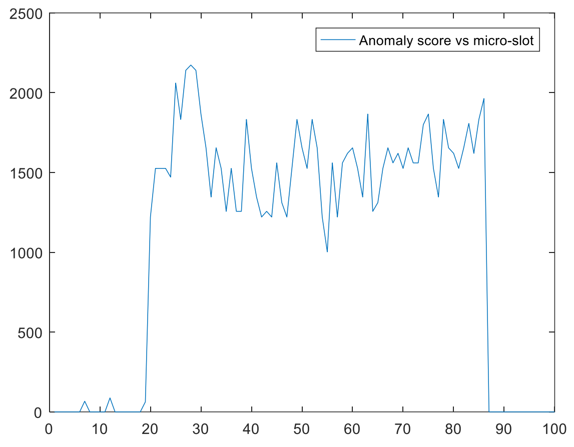

Figure 5. Plot of the abnormality score, with DDOS attack occurring at 20th micro-slot

As shown, the range of the abnormality score when there is a DDOS attack is much higher (more than in a 10x ratio) than the range of the abnormality score when there is no DDOS attack. The simulation also identifies the subspace (y_1, y_2) (TCP SYN ratio vs ICMP ratio) as the most discriminant subspace for this data set. It also finds the source addresses of DDOS flows. We have checked in the Wireshark logs that these flows are rogue flows (they all are unfinished TCP connection requests).

This algorithm has been implemented and fine-tuned on a prototype. Most of the algorithm functions are encoded from the MATLAB with MATLAB encoder.

5. PROTOTYPE RESULTS

5.1 Setup description

The experimental set-up (figure 6) consists of 2 Linux PCs. The CPU of these PCs is an Intel core i5 chipset at 2.3GHz. The 1st PC simulates the IVC-U and the In-Vehicle Infotainment (IVI) system. The 2nd PC simulates the Internet services or the server where car data are uploaded and the DDOS attack is hosted. They both are inserted in a local network. The 1st PC hosts a VLC server and the algorithm implementation. Its IP address is 192.168.0.1. The 2nd PC reads the video which is streamed by the 1st PC. The IP address of the VLC client is 192.168.0.2. The DDOS attack is a TCP SYN flood attack: it simply sends a TCP SYN (which is never acknowledged) on IP address 192.168.0.3. The TCP port index is incremented after each sent TCP SYN packet so that the TCP ports are scanned until the port index wraps around.

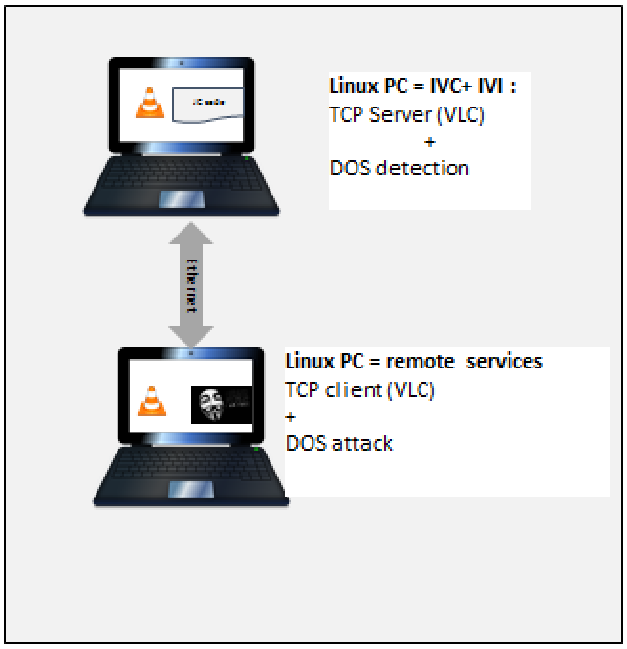

Figure 6. Prototype setup.

5.2 Measured abnormality score

The parameters of the algorithm must be tuned so that the abnormality score is as low as possible when there is no DDOS attack. The key parameter is the observation window size. If it is too small, it can't grab a wide variety of incoming data traffic. A new type of legitimate data traffic could result in a peak of abnormality score and it could result in a false positive. Figure 7 illustrates this.

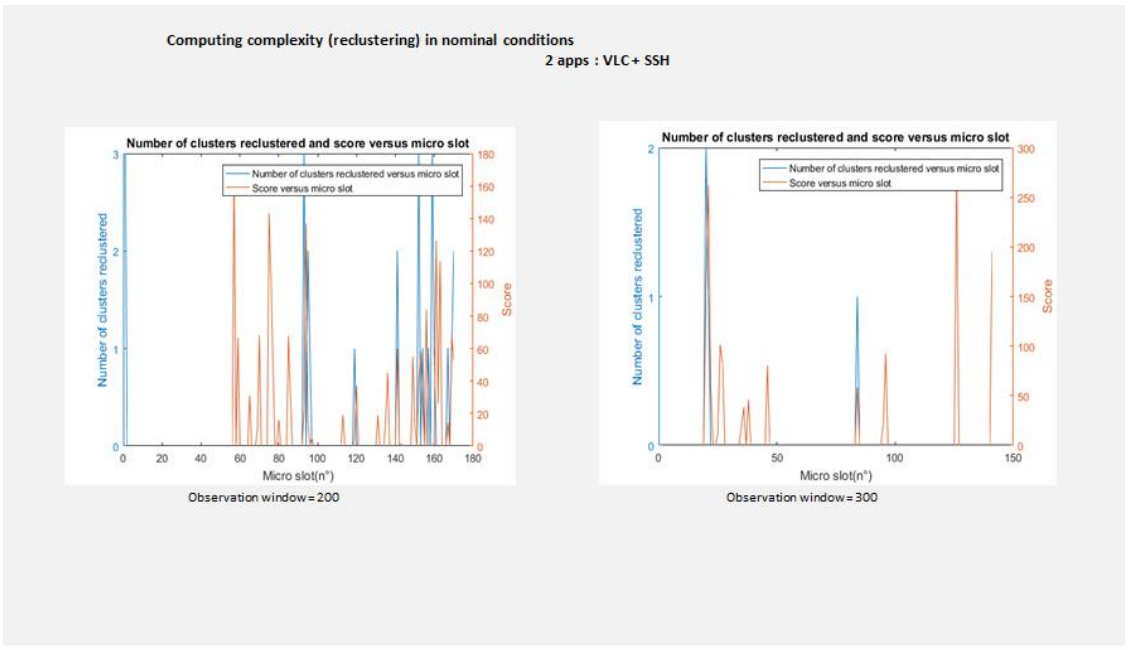

Figure7. Abnormality score without any DDOS attack for observation window size = 200 and 300

The brown curve plots the abnormality score. A SSH application is randomly triggered in parallel with the VLC server. The SSH features are different from the features of the VLC traffic. When the observation window size = 200, the SSH application causes some peaks of abnormality score. When it is 300, there is a higher probability that features vectors of the previous SSH are present in the observation window. Therefore, when a new SSH is started, there is a high probability that the A matrix of the observation window has at least one feature vector close to the new input vector. That is the reason why the peaks of abnormality score are much less frequent with an observation window = 300 than with an observation window = 200.

The blue plot of figure 7 also shows the number of subspaces to be repartitioned. It shows that the occurrences when re-clustering of the 2D-subspaces is needed are indeed very scarce. It translates to the fact that the algorithm does not need high computing power. With an observation window = 300, the maximum number of subspaces to be re-clustered is 2 whereas it is 3 with an observation window = 200. The occurrences when re-clustering shall be done is also less frequent.

The prototype has been tested and evaluated with an observation window = 300.

The abnormality score is measured when the DDOS occurs. It is shown in figure 8. For this test, the DDOS filtering stage of the algorithm is disabled.

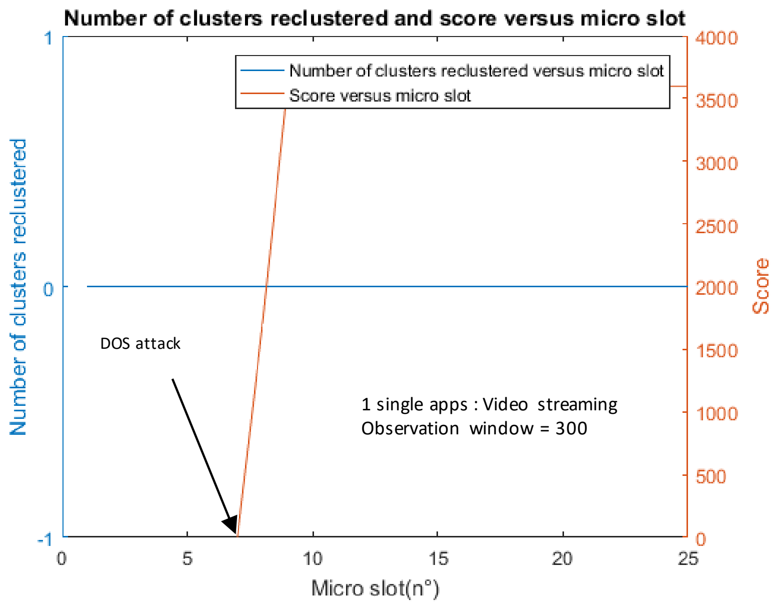

Figure 8. Abnormality score when a DDOS attack occurs

When the DDOS attack occurs, the abnormality score jumps from close to 0 to more than 3500. It is much more than the abnormality score peaks of figure 7, which are caused by a transient change of the type of incoming data traffic. Therefore, it is easy to identify DDOS attacks.

When the DDOS filtering stage is enabled, the IP address of the attack is identified and blacklisted. A new VLC connection can be re-established.

5.3 Measured CPU load of the proposed algorithm and performance discussions

The CPU load has been measured with the Linux Top command. The 1st measurement has been done when there is no DDOS attack: it measures the CPU load penalty of the algorithm to monitor the incoming traffic and search a DDOS pattern. The CPU is an Intel core i5 chipset at 2.3GHz.

Table 1. CPU load breakdown when there is no DDOS attack

VLC process	5.5%
DDOS detection & filtering algorithm	<0.3%

Thus, the penalty of the algorithm in terms of incremental CPU load when there is no DDOS attack (which is the case in 99% of the IVC-U lifetime) is very low. The incremental CPU load is less than 1% for this scenario. Such a result is consistent with the fact that the clustering step itself occurs very rarely and only for very few subspaces. The CPU load of the algorithm is also much less than the CPU load of the applications (in about a 20x ratio for this test). Therefore, it should not degrade the quality of service and it should be compatible with the computing capability of the IVC-U. Therefore, the main goal of this algorithm is met.

A 2nd measurement has been done when there is a DDOS attack but the DDOS filtering stage is disabled.

Table 2. CPU load breakdown when there is a DDOS attack but the DDOS filtering stage is disabled

VLC process	5%
DDOS detection & filtering algorithm	2%

A 3nd measurement has been done when there is an DDOS attack but the filtering stage is enabled: it measures the CPU load penalty of detecting the DDOS attack, identify the rogue IP flows and filter them out.

Table 3. CPU load when there is a DDOS attack with filtering stage enabled

VLC process	5%
DDOS detection & filtering algorithm	2.3%

In all these 3 scenarios, the relative CPU load increase is less than 2.5% and less than the main application. That is the reason why we think that the CPU penalty of this algorithm to detect and eliminate DDOS attacks is reasonable and is compatible with the computing power of an IVC-U.

Most of the papers which can be found in the literature are focused on the intrinsic performances of the algorithms: what is the false alarm rate? How much is it capable of detecting a DDOS attack? These algorithms are not integrated in embedded systems where the cost is as critical as performances. They do not really care about the best trade-off cost vs performances. So far, we have not found any paper which is focused on finding this optimal trade-off and which published some data about the incremental CPU load which is needed to detect DDOS attacks and filter these attacks.

Regarding the performances themselves, the automotive car manufacturers do not transmit safety-critical or privacy-sensitive data over the public Internet network. These data are carried over the private car network. Therefore, the key performance which matters here is the false alarm rate. If a DDOS attack is not detected, it is only impacting the infotainment applications. In this context, we have not detected any false alarms after several hours of testing where the incoming data traffic is a mix of multimedia flow (which is the typical data traffic over an IVC-U). All the TCP SYN Flood attacks we triggered have been detected.

However, in the near future, with the deployment of 5G Edge Computing, more and more services for connected and cooperative mobility will be proposed to enhance vehicle traffic efficiency and driver safety. These services will be even mandatory for the high level of autonomous driving (AD level 4 or more). It means that multiple cellular connections which carry safety-critical data will be open and these connections can suffer of the DDOS attack. Thus, the True Positive Rate KPI becomes very crucial. But an IVC-U is not network equipment. The True Positive Rate (TPR) and False Positive Rate (FPR) performances shall be assessed while considering the specific features of the received data traffic of these future services.

6. Conclusion and Next Steps

The proposed algorithm to detect and filter DDOS attacks on an IVC-U relies on the fact that the algorithm works in 2 steps:

- The incoming traffic is aggregated to build a single feature vector; this feature vector is used to detect a DDOS pattern as an outlier with regards to the clusters which have been built during an observation window of legitimate received traffic,
- When a DDOS attack is detected, the legitimate flows and the DDOS flows are in split by using only the most discriminant subspace and the flows whose anomaly score in this subspace is high are filtered out.

This approach allows to minimizing the requirements of CPU computation power. As shown, when there is no DDOS attack (which should be about 99% of the lifetime of the IVC-U), the additional computation power which is needed to monitor the incoming traffic is very low and does not degrade the quality of service of the active services. Since the identification of the DDOS flow is done only in one subspace (thanks to the 1st step), it also allows minimizing the peak needs of CPU computation power when an DDOS attack appears. Thus, it is compatible with low-cost CPUs of an IVC-U.

Further studies are needed to complete this work. In particular, it is key to analyze how much the grid size, the density threshold, and the attack detection threshold impact the True Positive Rate and False Positive Rate performances of DDOS attack detection. Since these parameters are likely correlated and jointly impact the performances, we must make sure that the mathematical method used to solve this problem seeks for the global optimum and not a local optimum.

The second next step would consist in adaptive algorithms for setting the detection threshold and the filtering threshold. The TPR and FPR performances would likely improve if these thresholds are adjusted depending on the type of applications involved in the received data traffic. But it has been demonstrated.

The last next step should consist in building a data test set which reflects the features of the received data traffic over the public Internet network and from providers of new cooperative and connected mobility services. Once done, we should be able to assess the True Positive Rate and False Positive Rate performances with a relevant data test bed.

ACKNOWLEDGEMENTS

The author wants to sincerely thank Philippe Owezarski (CNRS-LAAS) for his support and his help to start this project and the time he took to explain the details of the Orunada algorithm.

REFERENCES

[1] Andrew Ng – Coursera Online – Machine Learning

[2] D. Ippoliti and X. Zhou. Online adaptive anomaly detection for augmented network flows. In IEEE 22nd Int. Symp. on Modelling, Analysis Simulation of Comput. and Telecommun. Syst., pages 433–442, Sept. 2014.

[3] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In Conf. on Applications, Technologies, Architectures, and Protocols for Comput. Commun., pages 219–230, New York, NY, USA, 2004. ACM.

[4] Juliette Dromard, Gilles Roudiere, Philippe Owezarski. Online and scalable unsupervised network anomaly detection method – IEEE Transactions on Networks and Service Management ,pages 34-47, 2017.

[5] N. Chen, A. Chen, and L. Zhou. An incremental grid density-based clustering algorithm. Journal of Software, 13(1), Aug. 2002.

[6] P. Casas, J. Mazel, and P. Owezarski. NETWORKING 2011: 10th Int. IFIP TC 6 Networking Conf., chapter UNADA: Unsupervised Network Anomaly Detection Using Sub-space Outliers Ranking, pages 40–51. Springer Berlin Heidelberg, 2011.

[7] Y. Bengio, O. Delalleau, and N. Le Roux. The curse of highly variable functions for local kernel machines. In Proc. of the 18th Int. Conf. on Neural Information Processing Systems, pages 107–114, Cambridge, MA, USA, 2005. MIT Press.

[8] Chiadighikaobi Ikenna Rene and Johari Abdullah. Malicious Code Intrusion Detection using Machine Learning And Indicators of Compromise. International Journal of Computer Science and Information Security (IJCSIS), Vol. 15, No. 9, September 2017

[9] A. Patcha and J. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. Comput. Netw., 51(12), Aug 2007.

[10] Vishwas S. Ciza, T. and Balakrishnan N. Usefulness of darpa dataset for intrusion detection system evaluation, 2014

[11] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski. Taming the Curse of Dimensionality in Kernels and Novelty Detection, pages 425– 438. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006

[12] M-LShyu,S-CChen,K.Sarinnapakorn,andL.Chang. A novel anomaly detection scheme based on principal component classifier. IEEE Foundations and New Directions of Data Mining Workshop, pages 171– 179, 2003.

[13] J. Dromard, V. Baudin, P. Owezarski, A. Mozo-Velasco, B. Ordozgoiti, and S. Gomez-Canaval.D4.3: Experimental evaluation of algorithms for online network characterizations. Technical report, FP7 European ONTIC project: Online Network Traffic Characterization, January 2017.

[14] J. Mazel. Unsupervised network anomaly detection. PhD thesis, INSA Toulouse, France, Dec. 2011

[15] Vijaya P. Vipin, D., Sattvik S., Sreevathsan S., and Gireesh K. T. Network intrusion detection system based on machine learning algorithms. International Journal of Computer Science and Information Technology (IJCSIT), 2(6):138–151, 2010.

[16] Deshmukh S. Patil, R. and Rajeswari K. Analysis of simple k-means with multiple dimensions using weka. International Journal of Advanced Research in Computer Science and Software Engineering, 110:14–17, 2015.

[17] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In Twenty-Eighth Australasian Comput. Science Conf. (ACSC2005), pages 333–342, 2005.

[18] P. Casas, J. Mazel, and P. Owezarski. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. Comput. Comm., 35(7):772 – 783, 2012.

[19] 11. L. Parsons et al., "Subspace Clustering for High Dimensional Data: a Review", in ACM SIGKDD Expl. Newsletter, vol. 6 (1), pp. 90-105, 2004 ACM SIGKDD Expl. Newsletter, vol. 6 (1), pp. 90-105, 2004

AUTHOR

Eric Perraud works at Renault Software labs in the Strategy Office. He had previously been working at Intel and Motorola as modem chief architect. He has a Telecommunication master from Sup Telecom Bretagne and a opto-electronics PhD from Sup-Aero